

SuperMAM User Guide
Version 1.0

Publish Date: 8/1/2017

Authors: Matthew Smith, Michael Spallino, Andy Tse

Table of Contents

Introduction	2
Construction	2
Hardware Requirements	2
Assembly	3
Software Installation	7
Configuration	8
Creating an Image	8
Creating an /etc/hosts file	9
NFS configuration	9
SSH keys	10
Torque configuration	10
Torque Server (login-node)	11
Torque Client (compute-#)	12
Creating a Default Queue	12
Setting a Submission Host	12
Software Development	13
Hello World in Fortran	13
Hello World in C	13
Compilation	14
Job Submission Script	14
Submitting a Job and Checking it's status	15
What kinds of problems can be Parallelized / Distributed	17
Solving the Integral of $f(x)=x^2$	18
Testing	20
Maintenance	23
User account administration	23
Physical Hardware Maintenance	24
Removing Stuck Jobs	24
Using mail	25
Restarting the Cluster	26
IP Tracking Tool	27
Setting Static IP addresses	28
Copying Files to the Cluster Over the Network	28
Appendix	29
Useful Links	29

Introduction

High Performance and Distributed Computing are extremely relevant hot topics in the Computer Science field today. We live in an age of astronomical amounts of data that need to be processed in the new ways and methodologies that come out of these research areas. Many labs are utilizing supercomputers to process data faster and perform extremely complicated calculations. SuperMAM intends to bring an affordable education option to students interested in supercomputer construction, maintenance, and software development. Before attempting a project like this, students should be familiar with UNIX, fortran, C. They should be vaguely familiar with parallel programming and distributed computing.

Construction

Hardware Requirements

For the initial development of a Raspberry Pi supercomputer, the following list of components are required. The number of compute nodes determines the value of n .

Hardware Component	Minimum Quantity	Purpose
Raspberry Pi 3	n	Main computation device for the supercomputer.
Network Switch (min 4 port)	$m/n *$	Facilitates communication between devices over a network
USB power bank (min 2.4 amp)	$m/n *$	Provides power for the Raspberry Pis
Cat5 ethernet cables	n	Connecting the Raspberry Pis to the switch
USB Micro B to USB A (M to M)	n	Connecting the Raspberry Pis to the power bank
Micro SD card (min 8gb)	n	Storage for the Raspberry Pis
Case rack	$m/n *$	Ensure that the Raspberry Pis are secure and safe from harm

Table I. Necessary Hardware Components

* m/n means the number of devices supported divided by the number of Raspberry Pis in the cluster.

Assembly

Setting the Raspberry Pis up with the Raspian Operating System, powering them on, and connecting them to the switch can be done with a few simple steps. Figure 1.2.1 shows the cluster connected to the switch and powered on. A network diagram is also shown in Figure 1.2.2.

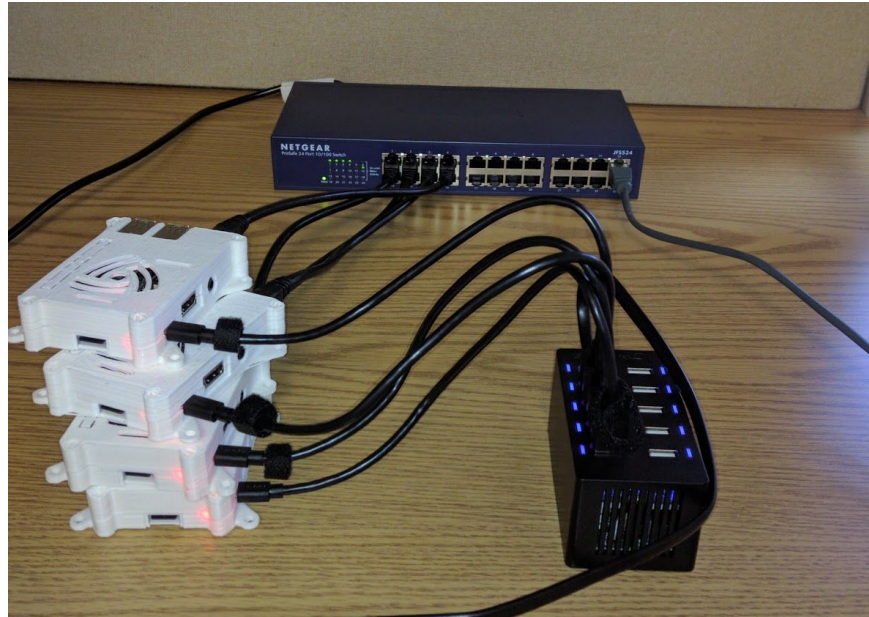


Figure 1.2.1 Full Cluster Assembly

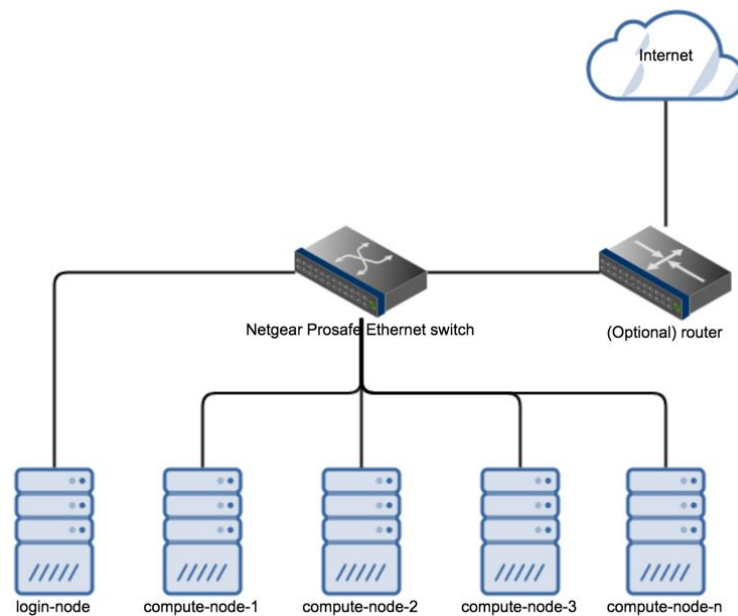


Figure 1.2.2 SuperMAM Network Diagram

Figure 1.2.3 and Figure 1.2.4 are included as reference images to show how components connect and what is housed in each case.



Figure 1.2.3 Exposed Raspberry Pi



Figure 1.2.4 Single Raspberry Pi Cluster

The Raspberry Pis should use the Raspbian Jessie Lite Operating System as the Desktop UI will not be required. The MicroSD card will need to be flashed with the Raspbian Jessie Lite Operating System with a program capable of burning disk images such as Etcher. Figure 1.2.5 shows where the MicroSD slot is on the Raspberry Pi.



Figure 1.2.5 Raspberry Pi MicroSD location

Links to downloads for Raspbian and Etcher can be found in the **Appendix: Useful Links** section.^{[8][9]} Figure 1.2.6 shows the Etcher User Interface when flashing the disk image.

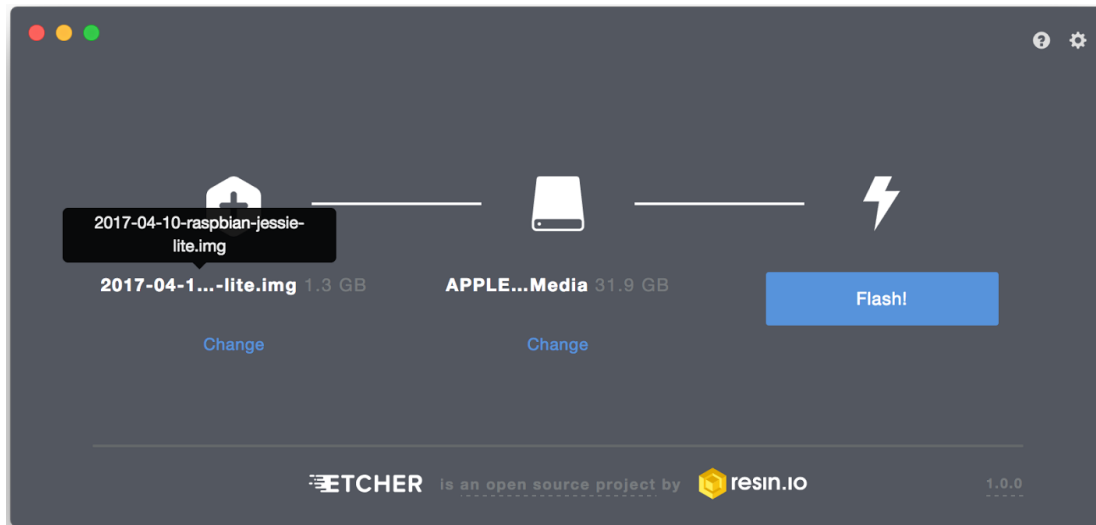


Figure 1.2.6 Flashing a MicroSD card in Etcher

When setting up a Raspberry Pi for the first time a monitor must be used to connect to the Raspberry Pi via HDMI. Once the Raspberry Pi boots there is a login. The following are the login credentials.

Username: `pi`

Password: `raspberrypi`

It is highly recommended that the password on this account be changed before SSH is enabled or removed entirely. User account administration is covered in the **Maintenance: User Administration** section.

Raspbian disables SSH by default, so it must be re enabled. The `raspi-config` command may be used to accomplish this. Select `Interface Options` from the menu and then select `SSH`. Pick `Yes` and select `Okay`. Finally select `Finish` to exit the configuration tool.^[10] Figure 1.2.7 and Figure 1.2.8 show these menus.

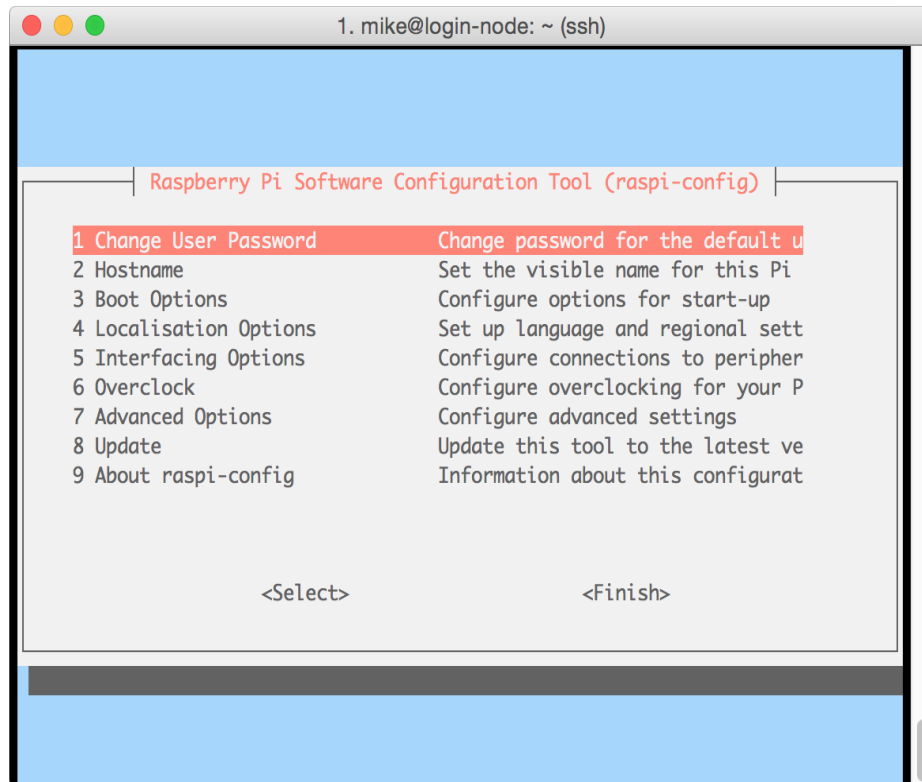


Figure 1.2.7 raspi-config main menu

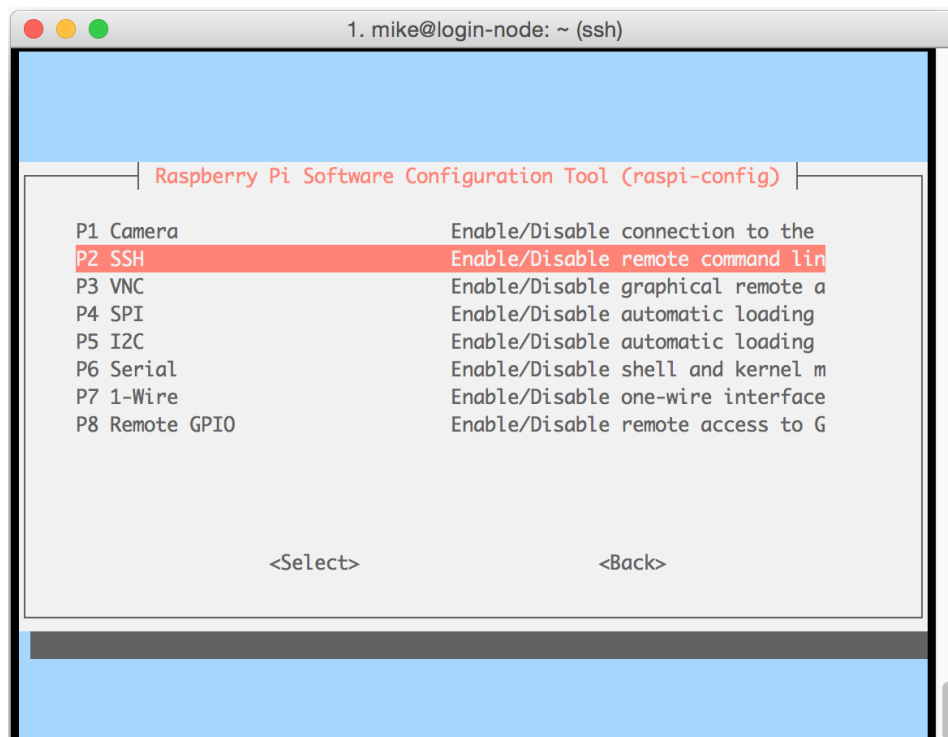


Figure 1.2.8 Interface Options in raspi-config

Software Installation

Be sure to run the following before installing the software packages listed in Table II which will require internet access.

```
$ sudo apt-get update
```

Software Components	Purpose
torque-server	The server used by torque
torque-client	The client used by torque
torque-mom	The server used by computational nodes to pull work off the queues in torque
torque-scheduler	Divides work for jobs across the cluster of compute nodes
nfs-kernel-server	Used to create a folder that other compute nodes can mount
nfs-common	Used to mount a folder from the login node on the compute nodes
gfortran	Compiler for fortran
openmpi-common	Library for message passing between nodes
openmpi-bin	
libopenmpi-dev	

Table II. Necessary Software Packages

The packages in Table II can be installed by executing:

```
sudo apt-get install package_name
```

Configuration

Creating an Image

The easiest way to bring up the compute nodes is to create an image with all of the software already installed. An image can be taken at any point and this may prove more useful after some of the following configurations are made. Rather than going through all of the following

configurations every time a new node must be brought up, an image can be taken after and then used instead of the base Raspbian Jessie Lite image used earlier.

To clone an image on Windows, download and install Win32 Disk Imager. When using Win32 Disk Imager, provide the path and filename of the future image in the Image File box. Select the SD card with the image from Device. Afterwards, click on Read to clone the image from the SD card to the Image File specified earlier. Use Etcher to clone the images onto other SD cards.

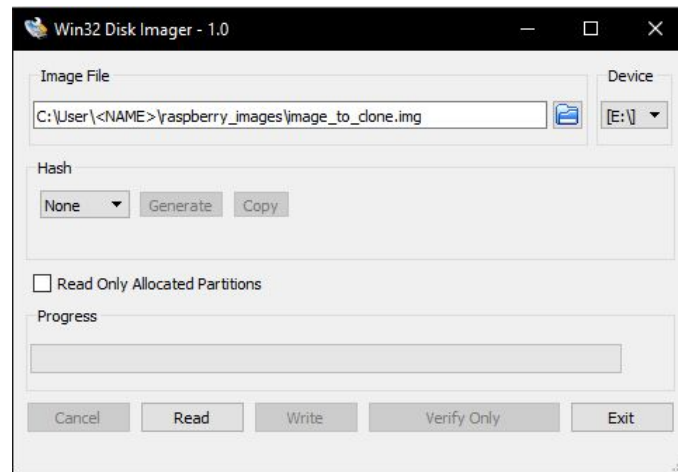


Figure 1.4.1 Win32 Disk Imager

To clone an image on Mac, open up Disk Utility. Highlight the SD card that requires cloning and create a New Image. Before saving the image, the image format must be DVD/CD master. After creating the image, change the extension of the image to .iso before using Etcher for the cloning process.

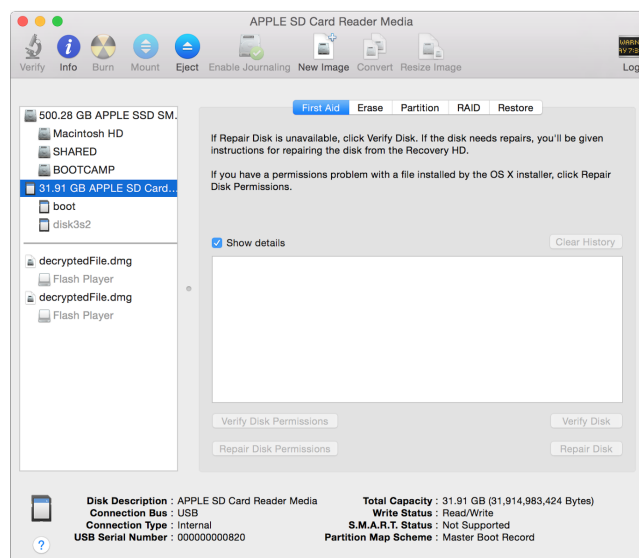


Figure 1.4.2 Disk Utility

Creating an /etc/hosts file

A hosts file will make it easier for navigating between the Raspberry Pis rather than using the ip addresses all of the time. The Torque and NFS configurations rely on hosts names configured here, so this step is important! The following is a sample `/etc/hosts` file:

```
127.0.0.1      localhost
::1           localhost ip6-localhost ip6-loopback
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters

10.14.50.20    login-node
10.14.50.114   compute-1
10.14.50.115   compute-2
10.14.50.41    compute-3
```

NFS configuration

Configuration of NFS is done through the file: `/etc/exports`.^[15] The following is a sample `/etc/exports` file.

```
/nfs/supermam  compute-1(rw,sync,no_subtree_check)
/nfs/supermam  compute-2(rw,sync,no_subtree_check)
/nfs/supermam  compute-3(rw,sync,no_subtree_check)
```

This directory can then be mounted by the compute nodes with the following command:

```
$ sudo mount login-node:/nfs/supermam /nfs/supermam
```

NOTE: When the Raspberry Pis are restarted, all of the compute nodes will need to have the directory remounted. The following commands may also be required to restart the NFS server on the login-node.

```
$ systemctl rpcbind start
$ systemctl nfs-kernel-server restart
```

SSH keys

The account(s) used for submitting jobs through torque must be capable of using SSH without a password to connect to the compute nodes. This can be achieved by creating an SSH key and adding it to the `authorized_keys` files of all of the compute nodes. Run the following command on the login node.

```
$ ssh-keygen -t rsa
```

Enter the default settings, don't change the directory or enter a passphrase. Then run the following command on the login node.

```
$ cat ~/.ssh/id_rsa.pub
```

Copy the output and then login to each compute node. Use vim or emacs to put the copied key in on a new line in `~/.ssh/authorized_keys` on the compute nodes.

This process must also be done from each compute-node to the login node because torque copies files back and forth between the hosts.

Torque configuration

This section describes what configurations are necessary for Torque to be used on the cluster. Before these configurations are set up, run the following commands on the login-node to set up a basic configuration ^{[6][7]}

If the torque-mom is running on the server, turn it off as it is not required for the server.

```
$ /etc/init.d/torque-mom stop
```

Stopping the scheduler and server are required so that the configurations are saved and used the next time torque runs.

```
$ /etc/init.d/torque-scheduler stop
$ /etc/init.d/torque-server stop
```

A server must be created to overwrite the old configuration and then killed to allow the new configurations to remain unaltered.

```
$ pbs_server -t create
$ killall pbs_server
```

Torque Server (login-node)

In this section and the following section the file name to edit is provided above each table and the contents of each table is what should be written to the file. In the following entries `compute-n` is meant to signify the `n`th compute node in the cluster. Four raspberry Pis means `compute-n` is `compute-4`.

```
/var/spool/torque/server_priv/nodes
```

```
compute-1 np=4  
compute-2 np=4  
compute-3 np=4  
...  
compute-n np=4
```

```
/etc/torque/server name
```

```
login-node
```

```
/var/spool/torque/server_priv/acl_svr/acl_hosts
```

```
login-node
```

```
/var/spool/torque/server_priv/acl_svr/operators
```

```
root@login-node
```

```
/var/spool/torque/server_priv/acl_svr/managers
```

```
root@login-node
```

Run the following command to configure scheduling of jobs:

```
$ qmgr -c 'set server scheduling = true'
```

This command may help prevent jobs from getting stuck in a queue:

```
$ qmgr -c 'set server mom_job_sync = true'
```

Torque Client (compute-#)

```
/var/spool/torque/mom_priv/config
```

```
login-node
```

Creating a Default Queue

A queue must be created in order to schedule jobs. The following commands set up a default queue named `batch` that uses `n` nodes where `n` is the size of the cluster.^[11]

```
$ qmgr -c "create queue batch queue_type=execution"  
$ qmgr -c "set queue batch started=true"  
$ qmgr -c "set queue batch enabled=true"  
$ qmgr -c "set queue batch resources_default.nodes=n"  
$ qmgr -c "set queue batch resources_default.walltime=3600"  
$ qmgr -c "set server default_queue=batch"
```

Setting a Submission Host

The following command sets the login-node as the only host that can submit jobs to the server.

```
$ qmgr -c 'set server submit_hosts = login-node'
```

Once all of the configuration is complete, the torque services may be restarted on the login-node and the compute nodes. On the server run:

```
$ /etc/init.d/torque-scheduler start  
$ /etc/init.d/torque-server start
```

On the compute nodes run:

```
$ /etc/init.d/torque-mom start
```

To ensure that all nodes are up and reachable by the login node run the following command on the login-node.^{[5][17][18]}

```
$ pbsnodes -a
```

Every node defined in `/var/spool/torque/server_priv/nodes` should be listed with a status other information about the systems.

Software Development

The best way to start learning a new system is to write a hello world program! The following two sections go over writing and running a hello world program in Fortran and in C.^[3]

In both examples the functions `MPI_INIT`, `MPI_COMM_SIZE`, `MPI_COMM_RANK`, `MPI_FINALIZE` are used. The `MPI_INIT` function initializes MPI for use. The `MPI_COMM_SIZE` and `MPI_COMM_RANK` functions are used to get the number of computation nodes in the cluster (size) and which node the program is executing on (rank). The `MPI_FINALIZE` function is used to close and cleanup MPI.^[19] The OpenMPI documentation contains a list of all the MPI functions, descriptions of what they do, and examples for C and Fortran.^[19]

Please note: With the cluster configuration described in this document, the login-node performs no computation. It is simply a job submission node. When running a job on the cluster on 3 nodes, one compute node will be rank 0, and the other two compute nodes will

be rank 1 and 2. The login-node will not participate in any computation unless it is configured to do so.

Hello World in Fortran

```
program hello
use mpi
  INTEGER rank, size, error_flag, tag, status(MPI_STATUS_SIZE)

  call MPI_INIT(error_flag)
  call MPI_COMM_SIZE(MPI_COMM_WORLD, size, error_flag)
  call MPI_COMM_RANK(MPI_COMM_WORLD, rank, error_flag)
  print*, 'node', rank, ': Hello world'
  call MPI_FINALIZE(error_flag)
end
```

Hello World in C

```
#include <stdio.h>
#include <mpi.h>

int main (argc, argv) {
  int rank, size;

  MPI_Init (&argc, &argv);
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  MPI_Comm_size (MPI_COMM_WORLD, &size);
  printf("node%d: Hello world", rank);
  MPI_Finalize();
  return 0;
}
```

Compilation

Programs that use MPI must be compiled with a different program. Instead of using `gfortran` or `gcc`, the programs `mpif90` and `mpicc` must be used to compile. These tools work basically identically to the originals. The following is an example of compiling the Hello World program for Fortran and C.^[2]

```
$ mpif90 hello_world.f90 -o hello_world_f
$ mpicc hello_world.c -o hello_world_c
```

Job Submission Script

Once the program has been compiled a submission script must be made to submit the job to a torque queue to be scheduled. The following is an example script that will run on one compute node in the cluster. The program executable `hello` for job was located in `/nfs/supermam`. This is the primary reason for the NFS being used. Rather than copying each executable to each compute node, the NFS folder will host that file for each compute node to point to .^[16]

`hello_submit.sh`

```
#!/bin/bash
# set the number of nodes and processes per node
#PBS -l nodes=1:ppn=1

# set max wallclock time
#PBS -l walltime=100:00:00

# set name of job
#PBS -N hello_world

# mail alert at start, end and abortion of execution
#PBS -m bea

# send mail to this address (not required)
#PBS -M user@login-node

# use submission environment
#PBS -V

# start job from the directory it was submitted
cd /nfs/supermam

# run through the mpirun launcher
mpirun -np 1 hello
```

In order to change the number of nodes the program runs on in the cluster, change the numbers after `nodes=` and `-np` in the above script. For example, running the hello world program on three compute nodes may be achieved by altering the lines mentioned as follows:

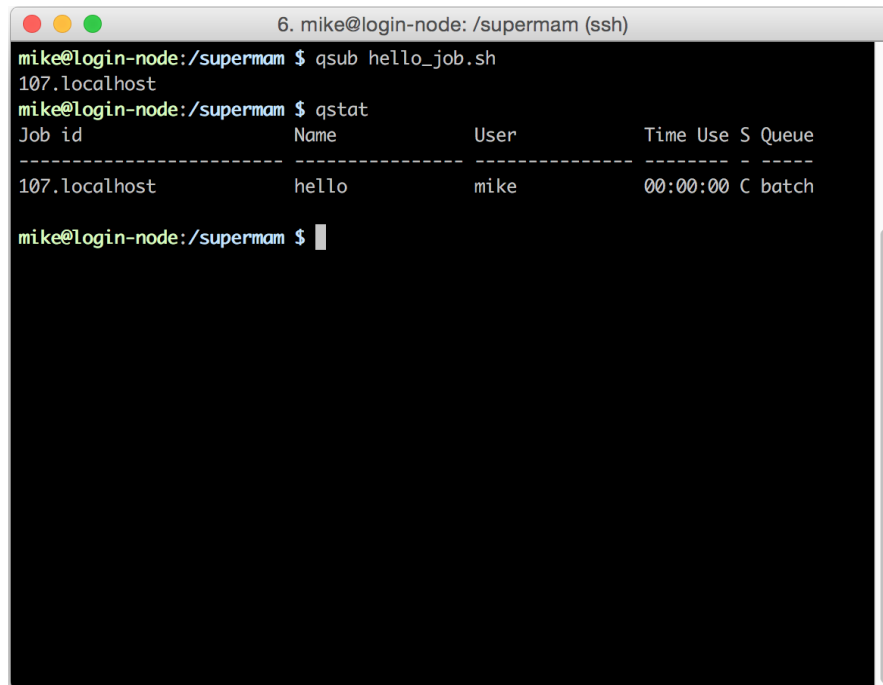
```
#PBS -l nodes=3:ppn=1
mpirun -np 3 hello
```

Submitting a Job and Checking it's status

Submitting a job can be done with the `qsub file_name.sh` command. The `file_name.sh` must be a job submission script detailed in the section above. The status of the job can be seen

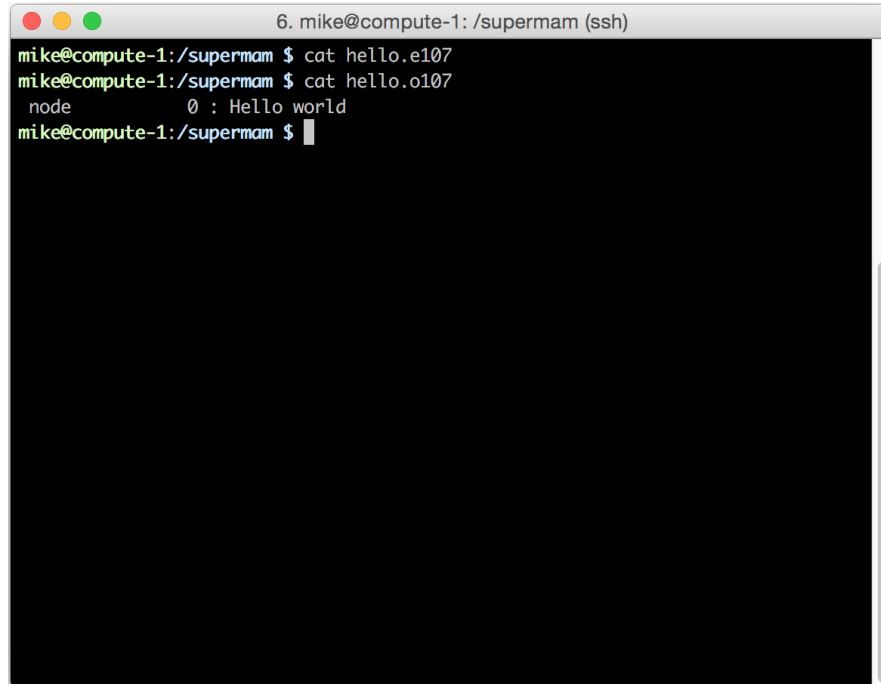
with the `qstat` command^{[5][7]}. Figure 2.4.1 should the result of `qstat` when running submitting the Hello World Fortran program.

When programs terminate, the produce two files in the following manner: `job_name.e#` and `job_name.o#`. The `.e` files contain any error messages produced and the `.o` files contain any output from the program execution. Figure 2.4.2 shows the contents of the output and error files generated for the Hello World Fortran job.



```
6. mike@login-node: /supermam (ssh)
mike@login-node:/supermam $ qsub hello_job.sh
107.localhost
mike@login-node:/supermam $ qstat
Job id          Name      User      Time Use S Queue
-----
107.localhost   hello     mike      00:00:00 C batch
mike@login-node:/supermam $
```

Figure 2.4.1 Result of the `qstat` Command

A terminal window titled "6. mike@compute-1: /supermam (ssh)". The prompt is "mike@compute-1:/supermam \$". The first command is "cat hello.e107", which produces no visible output. The second command is "cat hello.o107", which outputs "node 0 : Hello world". The prompt returns to "mike@compute-1:/supermam \$".

```
mike@compute-1:/supermam $ cat hello.e107
mike@compute-1:/supermam $ cat hello.o107
node      0 : Hello world
mike@compute-1:/supermam $
```

Figure 2.4.2 Output Files from a Job

What kinds of problems can be Parallelized / Distributed

It's important to note which kinds of problems can and can't be parallelized, because not all computations can be made faster this way. Problems that involve solving something iteratively where no solution impacts the solution at another successive iteration is something that can be parallelized.

A perfect example of this is solving an integral. Solving an integral can be approximated by using the rectangle method to find the sum of the areas of the rectangles under a curve. Each individual computation has no effect on the previous or future computation and these computations are performed over the range of the integral. Figure 2.5.1 shows left rectangular approximation with a rectangle width of 0.5. Obviously this is an underestimate of the area. However, the smaller the width of the rectangles, the solution will be closer to the true analytic solution for the integral of x^2 ($x^3/3$).

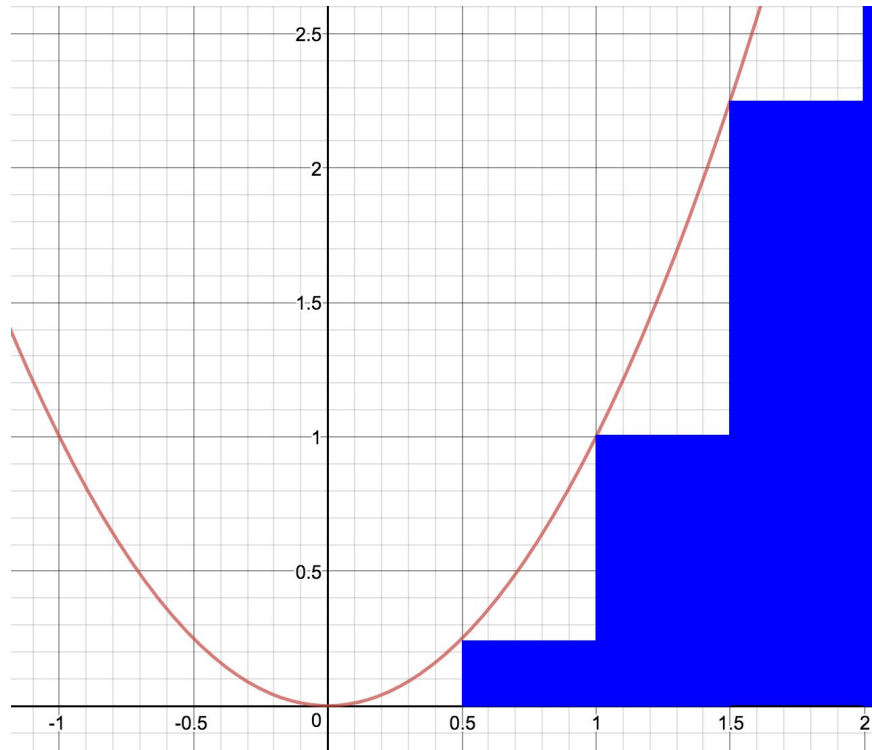


Figure 2.5.1 Left Riemann Sum of $f(x) = x^2$

Solving the $\int_0^{10} x^2 dx$ is trivial, as this is an easy integral to solve analytically, but the concept behind approximating the solution can be applied to more difficult integrals. There are some integrals where the analytical solution could be more difficult to find than just approximating it. The program given in the next section solves this integral from 0 to 10 with a rectangle width of 0.0001 and is an introduction on how to use the `MPI_Send` and `MPI_Recv` calls to break up work over the range of a problem.

Solving the Integral of $f(x)=x^2$

This section contains a program listing of a program to solve $\int_0^{10} x^2 dx$. Any other function may be substituted for `XSQUARED` to perform left rectangular approximation. Figure 2.6.1 shows the output of running this program over two nodes in the cluster.

```

program integral
use mpi

INTEGER rank, size, ierror, tag, status(MPI_STATUS_SIZE), processor, n, my_n, source
REAL (kind=8) a
REAL (kind=8) b
REAL (kind=8) new_a

```

```

REAL (kind=8) new_b
REAL (kind=8) my_a
REAL (kind=8) my_b
REAL (kind=8) total_area
REAL (kind=8) area
REAL (kind=8) wtime

interface INTEGRATE
  function INTEGRATE(a, b)
    REAL(kind=8) :: a
    REAL(kind=8) :: b
    REAL(kind=8) :: INTEGRATE
  end function
end interface

call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, size, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)

source = 0
a = 0
b = 10

if (rank == 0) then
  wtime = MPI_Wtime()
end if

if (rank == 0) then
  !Send the integral range
  do processor = 1, size - 1
    new_a = (((size - processor) * a) + (processor - 1) * b) / (size - 1)
    tag = 1
    call MPI_Send (new_a, 1, MPI_DOUBLE_PRECISION, processor, tag, MPI_COMM_WORLD,
ierror)

    new_b = (((size - processor - 1) * a) + (processor * b)) / (size - 1)
    tag = 2
    call MPI_Send (new_b, 1, MPI_DOUBLE_PRECISION, processor, tag, MPI_COMM_WORLD,
ierror)
  end do

  total_area = 0
  area = 0
else
  !Receive the integral range
  source = 0
  tag = 1
  call MPI_Recv (my_a, 1, MPI_DOUBLE_PRECISION, source, tag, MPI_COMM_WORLD, status,
ierror)

  tag = 2
  call MPI_Recv (my_b, 1, MPI_DOUBLE_PRECISION, source, tag, MPI_COMM_WORLD, status,
ierror)

  !Call integrate
  area = INTEGRATE(my_a, my_b)

  write ( *, '(a,i8,a,g14.6)' ) &
    ' Process ', rank, ' contributes area = ', area
end if

call MPI_Reduce(area, total_area, 1, MPI_DOUBLE_PRECISION, MPI_SUM, 0, MPI_COMM_WORLD,
ierror)

```

```

if (rank == 0) then
    wtime = MPI_Wtime() - wtime
    write ( *, '(a,g24.16)' ) ' Estimate = ', total_area
    write ( *, '(a,g14.6)' ) ' Time      = ', wtime
end if

call MPI_FINALIZE(ierr)

if (rank == 0) then
    write ( *, '(a)' ) ' '
    write ( *, '(a)' ) 'test integral:'
    write ( *, '(a)' ) ' Normal end of execution.'
end if
stop
END

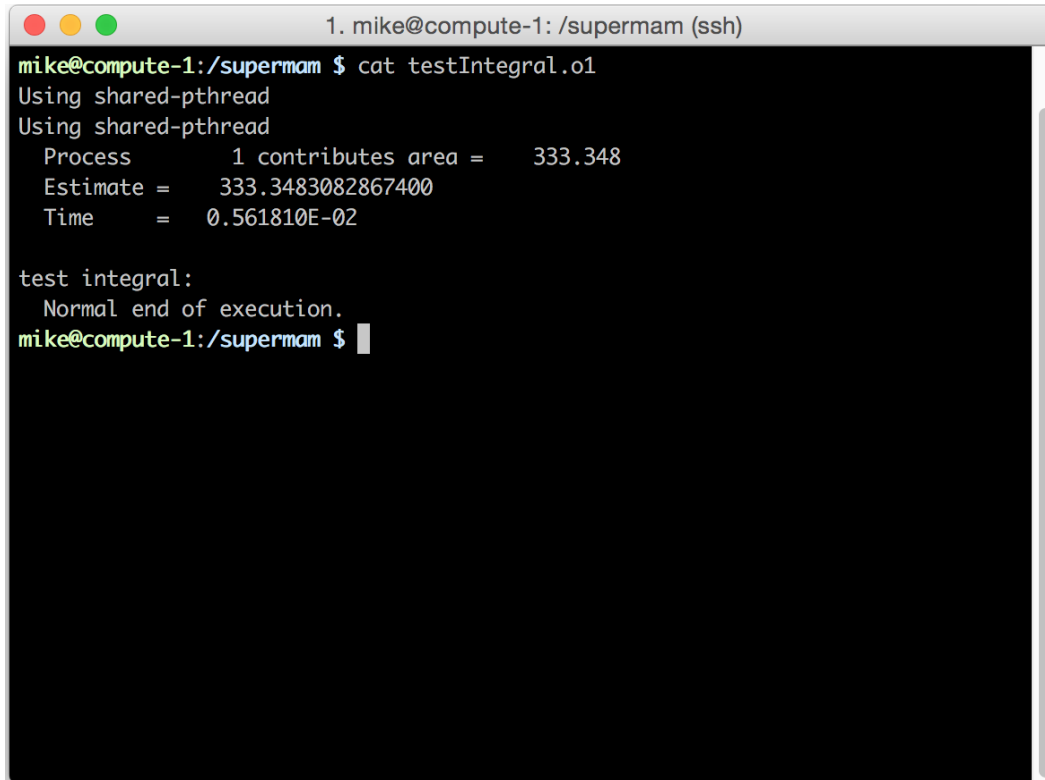
FUNCTION INTEGRATE(a, b)
    REAL(kind = 8) a
    REAL(kind = 8) b
    REAL(kind = 8) x
    REAL(kind = 8) y
    REAL(kind = 8) step
    REAL(kind = 8) INTEGRATE

    interface XSQUARED
        function XSQUARED(x)
            REAL(kind = 8) :: x
            REAL(kind = 8) :: XSQUARED
        end function
    end interface

    step = 0.0001
    x = a
    INTEGRATE = 0
    DO WHILE (x < b)
        x = x + step
        y = XSQUARED(x)
        INTEGRATE = INTEGRATE + (y * step)
    END DO
    RETURN
END FUNCTION

FUNCTION XSQUARED(x)
    REAL (kind=8) x
    REAL (kind=8) XSQUARED
    XSQUARED = x ** 2
    RETURN
END FUNCTION

```

A terminal window titled "1. mike@compute-1: /supermam (ssh)" showing the execution of a program. The user runs "cat testIntegral.o1", which displays the following output:

```
mike@compute-1:/supermam $ cat testIntegral.o1
Using shared-pthread
Using shared-pthread
Process      1 contributes area =    333.348
Estimate =   333.3483082867400
Time        =   0.561810E-02

test integral:
Normal end of execution.
mike@compute-1:/supermam $
```

Figure 2.6.1 Output of integral Program

Testing

The Tuning and Analysis and Utilities (TAU) may be used to profile code to figure out where most of the computation time is being spent during an execution. It can be installed with the following commands.^{[13][14]}

```
$ wget http://tau.uoregon.edu/pdt.tgz
$ tar -xvf pdt.tgz
$ cd pdtoolkit-#.#
$ ./configure
$ make && make install
$ wget http://tau.uoregon.edu/tau.tgz
$ tar -xvf tau.tgz
$ cd tau-#.#.#
$ export MPI_PATH=/usr/lib/openmpi
$ ./configure -mpilib=$MPI_PATH/lib -mpiinc=$MPI_PATH/include -mpi
-pdt=/PATH_TO_PDT/pdtoolkit-3.24
$ make install
$ export
TAU_MAKEFILE=/PATH_TO_TAU/tau-2.26.2p1/arm_linux/lib/Makefile.tau-mpi-pdt
```

Once Tau has been installed and configured, programs must be compiled with `tau_f90.sh` or `tau_cc.sh`. The following command is an example of compiling the `integral` program from the previous section.

```
$ tau_f90.sh integral.f90 -o integral_tau
```

These programs can still be run with MPI through the `mpirun` command. Once run, files named `profile.###` will appear in the directory where the job was submitted from. These files have profiling information from each machine that ran the code. These files can be read in a better text format with the program `pprof`.

```
$ pprof
```

This will location all of the `profile.###` files in the current directory and display the information in tables. A specific `profile.###` file may be passed to `pprof` to isolate the output to a specific profile of interest using the `-f` option and passing the filename. Figure 3.1.1 and Figure 3.1.2 show an example of a profile generated from running the `integral_tau` program.

These profiles are useful for determining which pieces of code are taking up the most computation time. When searching for performance optimizations it will be critical to identify which functions or subroutines are taking the most time, as there may be a leak or simple algorithmic optimization to speed up the program.

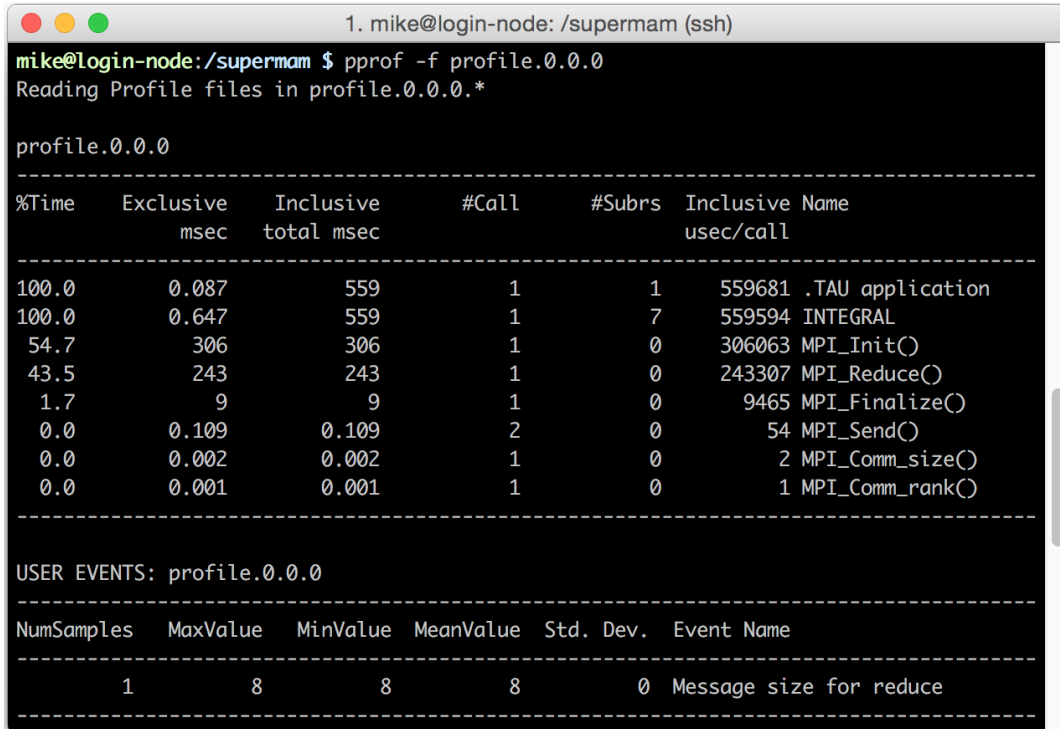


Figure 3.1.1 First Profile

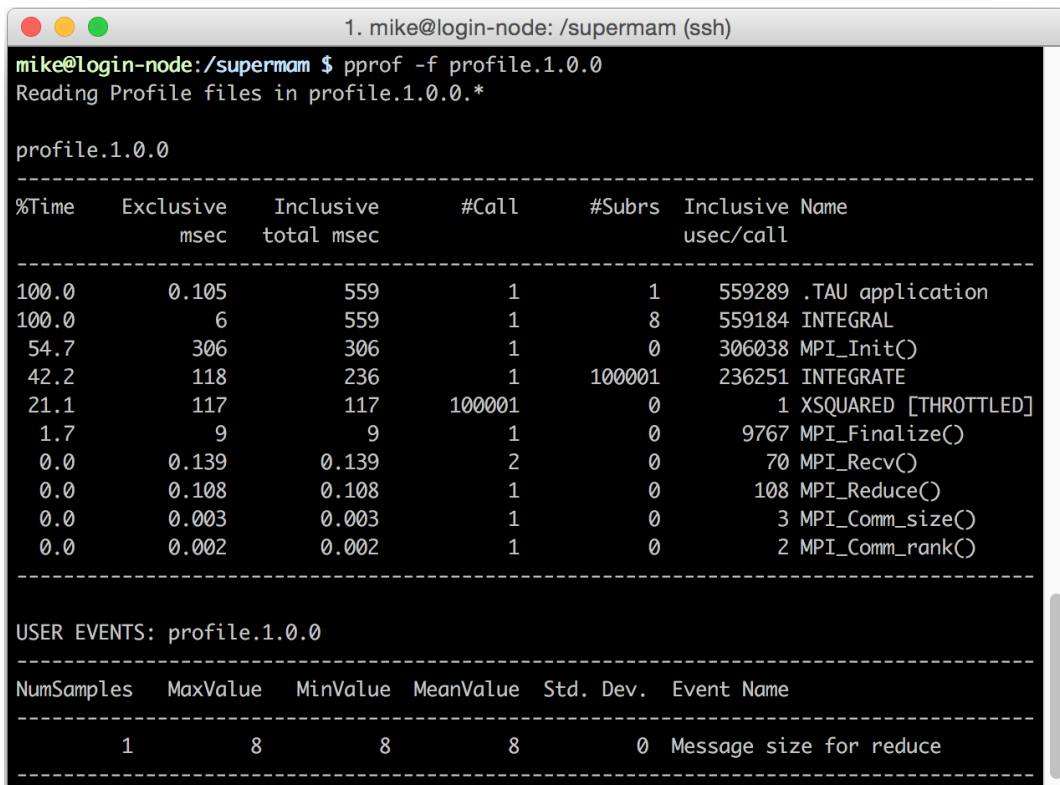


Figure 3.1.2 Second Profile

Maintenance

User account administration

NOTE: All of the following commands must be executed with superuser privileges.

Users may be created and changed with a few unix commands: `useradd`, `passwd`, and `usermod`.

Creating a new user can be done by executing:

```
$ useradd username_goes_here
```

That user's password can then be set with:

```
$ passwd username_goes_here
```

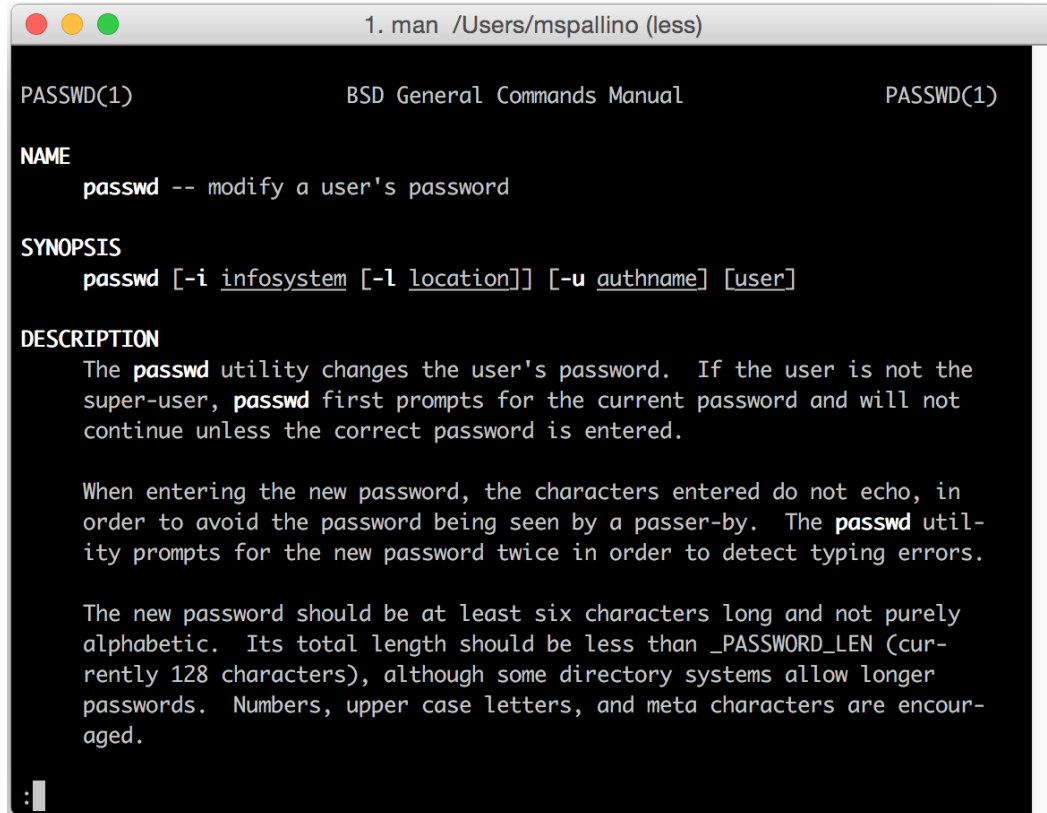
Rather than making every user a superuser, it is better to allow users to become root with the `sudo su` command. To do this execute the following:

```
$ usermod -aG sudo username_goes_here
```

If the user does not have a home directory `/home/username_goes_here` then one can be created by executing the following command:

```
$ mkhomedir_helper username_goes_here
```

The man pages for these commands are a good place to learn more about them. They can be read by executing `man command_goes_here`. An example of a man page can be seen in Figure 5.1.1 below.



```
1. man /Users/mspallino (less)

PASSWD(1)                                BSD General Commands Manual                                PASSWD(1)

NAME
    passwd -- modify a user's password

SYNOPSIS
    passwd [-i infosystem [-l location]] [-u authname] [user]

DESCRIPTION
    The passwd utility changes the user's password.  If the user is not the
    super-user, passwd first prompts for the current password and will not
    continue unless the correct password is entered.

    When entering the new password, the characters entered do not echo, in
    order to avoid the password being seen by a passer-by.  The passwd util-
    ity prompts for the new password twice in order to detect typing errors.

    The new password should be at least six characters long and not purely
    alphabetic.  Its total length should be less than _PASSWORD_LEN (cur-
    rently 128 characters), although some directory systems allow longer
    passwords.  Numbers, upper case letters, and meta characters are encour-
    aged.

:
```

Figure 4.1.1 Example man page from OSX for the passwd command

Physical Hardware Maintenance

The Raspberry Pis should be housed in cases or a rack to make sure they aren't touching one another. The Raspberry Pis can also get relatively hot while operational, so keeping them in a well ventilated area is highly advised.

Removing Stuck Jobs

If a job becomes stuck for some reason, like a deadlock, the following command may be used to stop the job.

```
$ qsig -s 0 <JOBID>
```

This command will tell the mom process to close. If that doesn't work, then the following command may be used, however, it is not recommended by Adaptive Computing.^{[5][12]}

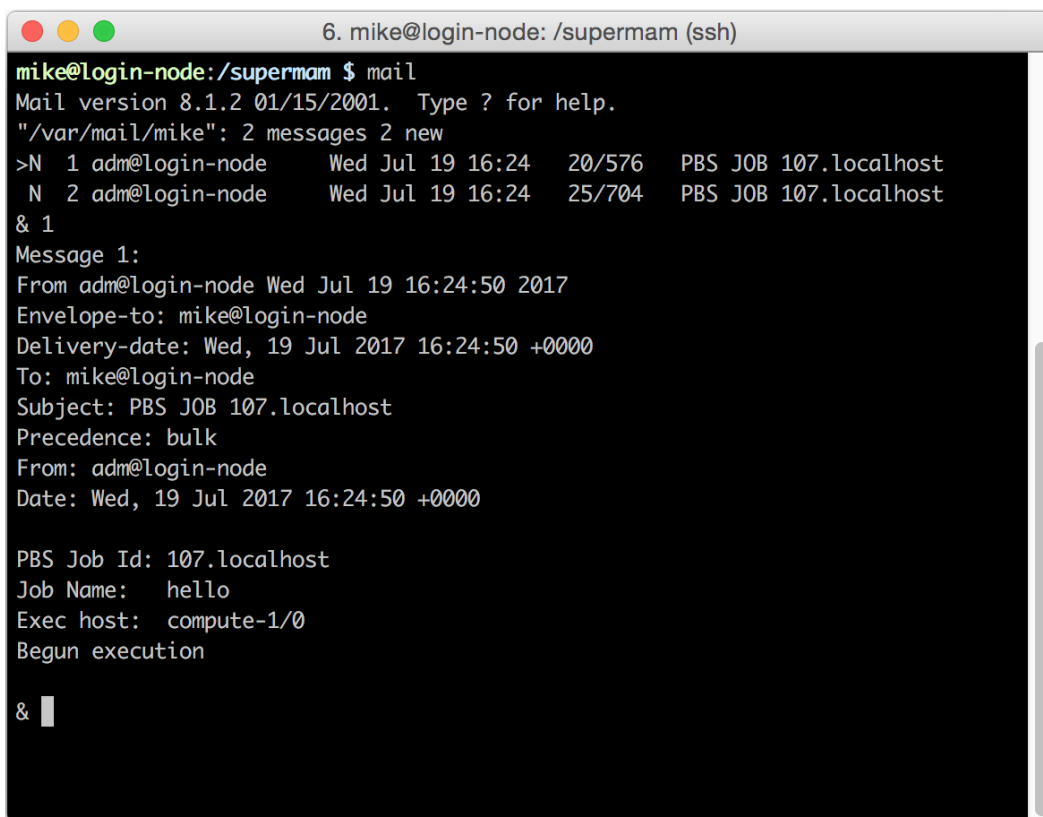
```
$ qdel <JOBID>
```

Using mail

Torque can send emails to a user when jobs start, complete, and error. The mail program can be run by executing the following command.

```
$ mail
```

Once inside the mail program, emails can be opened by typing the number of the email found in the second column and pressing enter. Figure 4.4.1 and Figure 4.4.2 show what the mail program looks like and two emails sent by torque concerning a submitted job.



```
6. mike@login-node: /supermam (ssh)
mike@login-node:/supermam $ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/mike": 2 messages 2 new
>N  1 adm@login-node      Wed Jul 19 16:24   20/576   PBS JOB 107.localhost
  N  2 adm@login-node      Wed Jul 19 16:24   25/704   PBS JOB 107.localhost
& 1
Message 1:
From: adm@login-node Wed Jul 19 16:24:50 2017
Envelope-to: mike@login-node
Delivery-date: Wed, 19 Jul 2017 16:24:50 +0000
To: mike@login-node
Subject: PBS JOB 107.localhost
Precedence: bulk
From: adm@login-node
Date: Wed, 19 Jul 2017 16:24:50 +0000

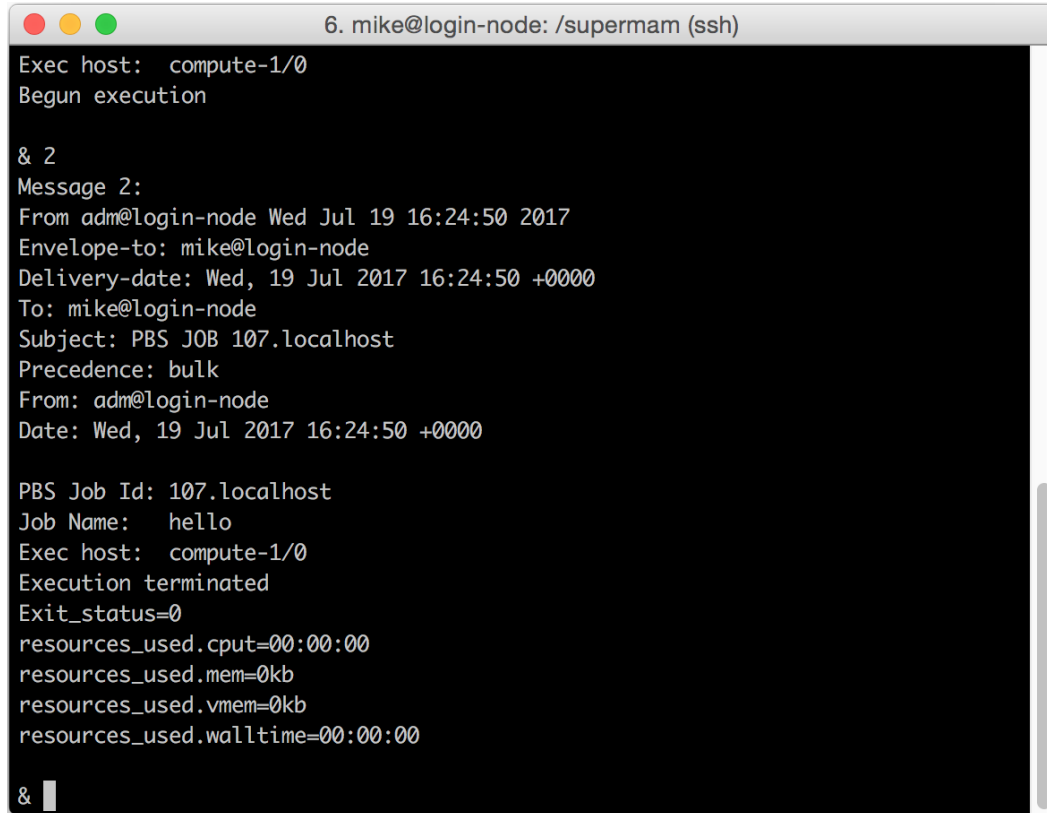
PBS Job Id: 107.localhost
Job Name:  hello
Exec host:  compute-1/0
Begun execution

& █
```

Figure 4.4.1 Using mail to Read Torque Emails

Pressing the `enter` key will also open new mail from the first new email to the last. The mail program can be quit by typing `q` and pressing `enter`.

Using mail can also be an easy way to communicate with other users on the cluster if the team is not colocated.



```
6. mike@login-node: /supermam (ssh)
Exec host: compute-1/0
Begun execution

& 2
Message 2:
From: adm@login-node Wed Jul 19 16:24:50 2017
Envelope-to: mike@login-node
Delivery-date: Wed, 19 Jul 2017 16:24:50 +0000
To: mike@login-node
Subject: PBS JOB 107.localhost
Precedence: bulk
From: adm@login-node
Date: Wed, 19 Jul 2017 16:24:50 +0000

PBS Job Id: 107.localhost
Job Name: hello
Exec host: compute-1/0
Execution terminated
Exit_status=0
resources_used.cput=00:00:00
resources_used.mem=0kb
resources_used.vmem=0kb
resources_used.walltime=00:00:00

& █
```

Figure 4.4.2 Using mail to Read Torque Emails

Restarting the Cluster

If the cluster needs to be restarted for some reason, there are a few commands that should be run to ensure that the cluster is still operational. The NFS mounts will no longer exist so the service must be restarted on the login-node and the compute nodes will need the directory re-mounted.

login-node:

```
$ service rpcbind restart
$ service nfs-kernel-server restart
```

compute nodes:

```
$ sudo mount login-node:/nfs/supermam /nfs/supermam
```

If the switch needs to be restarted, all of the IP addresses of the Raspberry Pis will likely change because the switch is unmanaged and will use DHCP to allocate new IP addresses. This is especially true if the switch is connected to the internet through some other network. To deal with this, the IP Tracking Tool mentioned below will prove useful. All of the `/etc/hosts` files will need to be updated because all of the configurations for the cluster use hostnames of the Raspberry Pis. If the switch is not connected to an external network and not connected to the

internet, setting static ip addresses for the pis as mentioned below will be possible rather than constantly editing the `/etc/hosts` file.

Check to make sure that the torque services are running everywhere.

login-node:

```
$ service torque_server status
$ service torque_scheduler status
```

compute-nodes:

```
$ service torque_mom status
```

If any are not running, restart them with the following command.

```
$ service SERVICE_NAME restart
```

IP Tracking Tool

The following script may be useful if the cluster is unplugged and the Raspberry Pis don't have static ip addresses configured. Get all of the mac addresses of each Raspberry Pi in the cluster and add it to the `macs` list. Set the value of `login` to the mac address of the login node. When this script is run on the login node, a new `/etc/hosts` file will be printed to the screen to be used. The program `nmap` may be used by itself to find the ip addresses of the Raspberry Pis, but this script recreates a host file in addition to using `nmap`.^[4]

```
#!/bin/bash
login="B8:27:EB:08:52:13"
macs=(B8:27:EB:C6:9A:90 B8:27:EB:0C:56:2E B8:27:EB:FC:B9:47)
count=1
tmp=""

echo -e "127.0.0.1\tlocalhost" > newhostfile.txt
echo -e ":::1\tlocalhost ip6-localhost ip6-loopback" >> newhostfile.txt
echo -e "ff02::1\tip6-allnodes" >> newhostfile.txt
echo -e "ff02::2\tip6-allrouters" >> newhostfile.txt
echo "" >> newhostfile.txt
echo -e "127.0.0.1\t`hostname`" >> newhostfile.txt

iprange=`hostname --all-ip-addresses |cut -d '.' -f -3`
nmap -sn `echo "$iprange.0/24"` > temp_nmap.txt
```

```

echo -e "`hostname --all-ip-addresses`\tlogin-node">> newhostfile.txt

for i in "${macs[@]}"
do
    tmp=`grep -B2 $i temp_nmap.txt | grep "Nmap scan" | cut -d " " -f 5`
    echo -e "$tmp\tcompute-$count" >> newhostfile.txt
    count=$((count+1))
done

cat newhostfile.txt
rm newhostfile.txt
rm temp_nmap.txt

```

Setting Static IP addresses

Edit `/etc/dhcpd.conf` on each of the Raspberry Pis and add the following lines to the end of each file with the desired unique IP address and subnet.

```

interface eth0

static ip_address=192.168.0.10/24

```

Copying Files to the Cluster Over the Network

It may prove useful to edit programs and other documents on a computer that is not a part of the cluster, especially if one is not comfortable with vim or emacs for text editing. If this is the case the scp program may be used to transfer files. The following is an example of how to copy a file named `my_program.f90` to the login-node `/programs` folder.

```
$ scp my_program.f90 user@login-node:/programs/.
```

Appendix

Useful Links

The following links were referenced a lot, especially when creating this document. If something is not clear here, these links may provide additional explanation.

[1] M. D. Schuster. Building a Raspberry Pi Supercomputer. 2017

[2] Compiling MPI programs:

https://www.dartmouth.edu/~rc/classes/intro_mpi/compiling_mpi_ex.html#top

[3] Hello World MPI examples:

https://www.dartmouth.edu/~rc/classes/intro_mpi/hello_world_ex.html

[4] Using nmap:

<https://www.digitalocean.com/community/tutorials/how-to-use-nmap-to-scan-for-open-ports-on-your-vps>

[5] Useful pbs commands for Torque:

<https://hpcc.usc.edu/support/documentation/useful-pbs-commands/>

[6] Setting up Torque clients:

<https://jabriffa.wordpress.com/2015/03/25/adding-client-nodes-to-a-torquepbs-system-on-ubuntu-14-04-lts/>

[7] Setting up a Torque server:

<https://jabriffa.wordpress.com/2015/02/11/installing-torquepbs-job-scheduler-on-ubuntu-14-04-lts/>

[8] Raspbian image download:

<https://www.raspberrypi.org/downloads/raspbian/>

[9] Etcher download:

<https://etcher.io/>

[10] Raspbian SSH instructions:

<https://www.raspberrypi.org/documentation/remote-access/ssh/>

[11] Torque Queue Configurations:

<http://docs.adaptivecomputing.com/torque/3-0-5/4.1queueconfig.php#default>

[12] Removing Stuck Jobs:

<http://docs.adaptivecomputing.com/torque/4-1-4/Content/topics/11-troubleshooting/stuckJobs.htm>

[13] TAU installation guide:

<https://www.cs.uoregon.edu/research/tau/tau-installguide.pdf>

[14] TAU By Example: https://wiki.mpich.org/mpich/index.php/TAU_by_example

[15] NFS Configuration:

<https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-16-04>

[16] Torque Submission Script:

<http://www.arc.ox.ac.uk/content/torque-job-scheduler#PBSubScript>

[17] TORQUE Administrator Guide: <http://docs.adaptivecomputing.com/torque/3-0-5/>

[18] TORQUE Admin Guide:

<http://docs.adaptivecomputing.com/torque/6-1-1/adminGuide/torqueAdminGuide-6.1.1.pdf>

[19] OpenMPI Documentation: <https://www.open-mpi.org/doc/current/>